


Persistência de Dados

Entender o que significa persistência é essencial para poder avaliar diferentes sistemas de armazenamento de dados. Como o armazenamento desempenha um papel crítico na maioria dos aplicativos modernos, fazer uma escolha imprudente de solução pode resultar em tempo de inatividade significativo ou perda de dados.

De acordo com SCHATZ, VIECELLI e VIGOLO (2023), persistência se trata da existência contínua de um efeito depois que sua causa foi removida. No que diz respeito ao armazenamento em um sistema de computador, isso significa que os dados persistem após o término do processo que os criou. Em outras palavras, um armazenamento é considerado persistente quando os dados são gravados no armazenamento persistente.

Existem várias formas de tornar os dados permanentes por meio de diferentes tipos de armazenamento permanente. Por exemplo, se estiver escrevendo um documento de texto, você pode salvar o documento em um arquivo e fechar o programa. Da mesma forma, criar uma cópia de backup do seu sistema ou arquivos é uma forma de persistência, pois permite recarregar o backup posteriormente e recuperar as informações. Alguns sistemas usam logs para registrar informações sobre persistência de dados.

Se você estiver escrevendo um programa de computador que usa persistência para armazenar dados, poderá usar os recursos internos de entrada e saída de arquivos da linguagem de programação para permitir que o usuário crie novos arquivos ou modifique os existentes. Se você precisar de uma solução de rede, por exemplo, ao criar um banco de dados, você pode programar formas de permitir que os usuários enviem

informações para as tabelas utilizando linguagem de consulta estruturada ou interfaces de programação de aplicativos.



Em algumas redes, mais de uma pessoa pode acessar dados persistentes, por exemplo, um banco de dados de rede. Porém, se o sistema não travar os dados de forma que no máximo apenas uma pessoa possa alterá-los, existe a possibilidade de duas ou mais pessoas tentarem fazer alterações, fazendo com que uma pessoa sobrescreva as alterações de outra pessoa. Um sistema com um método transacional permitirá que você saiba quando alguém fizer uma alteração que você está visualizando, antes de fazer qualquer outra alteração. Isso garante que você tenha acesso às informações mais atualizadas (JANDL JUNIOR, 2015).

Persistência em Banco de Dados Relacional

Um banco de dados relacional, também conhecido como RDB (Relational Database), caracteriza-se por gerenciar dados na forma de tabelas. Geralmente, possui várias tabelas para gerenciar os dados necessários para o funcionamento do sistema. Uma tabela consiste em dois elementos.

Existem muitos tipos de bancos de dados, e o formato para manipulação de dados difere dependendo do tipo. Entre os bancos de dados, o RDB deve ser considerado como tratamento de dados como tabelas, linhas (registros) e colunas (colunas). Conforme explicado por COELHO (2014), o conceito de tabela não é estritamente definido como um RDB, mas na prática será mais fácil de entender se você pensar nela como uma tabela. Ao extrair algumas dessas linhas ou colunas ou unir várias tabelas, os usuários podem realizar operações arbitrárias.

O software que gerencia o RDB nesse formato é chamado de RDBMS (sistema de gerenciamento de banco de dados relacional). Dentre os softwares disponíveis, podemos relacionar os seguintes: PostgreSQL e

MySQL em código aberto, e Microsoft SQL Server e Oracle Database, que possuem licenças comerciais.



Frameworks de persistência

Uma estrutura de persistência move os dados do programa em sua forma mais natural (em objetos de memória) de ou para um armazenamento permanente.

A estrutura de persistência gerencia o banco de dados e o mapeamento entre o banco de dados e os objetos. Existem muitos frameworks de persistência (tanto de código aberto quanto comerciais) no mercado. A estrutura de persistência simplifica o processo de desenvolvimento.

De acordo com JANDL JUNIOR (2015), podemos relacionar algumas estruturas de persistência de código aberto:

- **Hibernate:** é um poderoso serviço de consulta e persistência objeto/relacional de alto desempenho para Java. Esse framework permite que você expresse consultas usando SQL nativo ou critérios baseados em Java e consultas de exemplo. O Hibernate é agora a solução de mapeamento objeto/relacional mais popular para Java.

- **ObJectRelationalBridge (OJB):** é uma ferramenta de mapeamento Objeto/Relacional que permite persistência transparente para Objetos Java em bancos de dados relacionais. OJB foi projetado para uma grande variedade de aplicativos, desde sistemas embarcados até aplicativos rich client e arquiteturas multicamadas baseadas em J2EE.

- **Torque:** é uma camada de persistência. O Torque inclui um gerador para gerar todos os recursos de banco de dados necessários para sua aplicação e inclui um ambiente de execução para executar as classes geradas. Torque foi desenvolvido como parte do Turbine Framework.

- **TriActive JDO (TJDO):** é uma implementação de software livre da especificação JDO da Sun (JSR 12), projetada para suportar persistênc transparente usando qualquer banco de dados compatível com JDBC. O TJDO foi implantado e executado com sucesso em muitas instalações comerciais desde 2001.

- **pBeans:** é uma camada de mapeamento de banco de dados Objeto/Relacional (O/R). Ele foi projetado para ser simples de usar e totalmente automatizado. A ideia é que você economize tempo e esforço simplesmente concentrando-se em escrever classes Java e não se preocupando com a manutenção de scripts SQL correspondentes, esquemas baseados em XML ou mesmo com a geração de código.

- **Smyle:** fornece uma abordagem inovadora e pragmática para armazenamento de dados. É fácil de entender, conveniente de usar, mas poderoso o suficiente para aplicativos do mundo real.

Persistência de dados em dispositivos móveis

Ao desenvolver aplicativos como jogos digitais, plataformas de comércio eletrônico, aplicativos de mídia social ou pipelines de dados, você precisa considerar como os diferentes dados que você está gerando e coletando serão usados e armazenados (RABIN, 2012).

Todos os dados que precisam estar disponíveis para uso após a conclusão do processo de seu aplicativo devem ser armazenados no armazenamento persistente. Isso vai além de apenas manter as transações que ocorrem no seu projeto, como a pontuação final do usuário ao término de uma partida. Sua empresa gera dados primários altamente valiosos ao longo de seu ciclo de vida, que podem ter utilidade futura. Você precisa descobrir quais desses dados são mais valiosos e garantir que sejam mantidos, enquanto os dados que são necessários apenas temporariamente podem ser usados e descartados com segurança (SCHATZ, VIECELLI e VIGOLO, 2023).

O armazenamento persistente de dados pode se tornar caro à medida que você gera mais e mais dados e sua base de usuários cresce. Informaçõ inúteis também podem dificultar o gerenciamento, dificultando a análise. Decidir quais dados persistir e quais não persistir requer um planejamento cuidadoso. Por exemplo, ao lidar com dados de clientes e pipelines que consomem e formatam dados de várias fontes, os dados provavelmente passarão por transformação. Os dados nessas etapas intermediárias provavelmente não serão necessários novamente, enquanto os resultados finais precisarão ser armazenados em uma mídia persistente para que possam ser capitalizados no futuro.

Geralmente, os usuários finais não precisam se preocupar com armazenamento persistente. Os aplicativos de nível de consumidor fornecem meios apropriados para armazenar dados para uso contínuo, seja como um arquivo para aplicativos de desktop ou em uma plataforma online para aplicativos em nuvem.

De acordo com JANDL JUNIOR (2015), cada vez mais os desenvolvedores também são amplamente dispensados de implementar a persistência de dados em um nível baixo. A maioria das estruturas de aplicativos modernas fornece as ferramentas e bibliotecas necessárias para ler e gravar dados em uma variedade de back-ends de armazenamento não voláteis e voláteis. Se você estiver desenvolvendo seu próprio aplicativo e decidindo entre as opções de armazenamento, (por exemplo, as plataformas de comércio eletrônico geralmente oferecem diferentes back-ends de banco de dados para você escolher), é necessário considerar a natureza dos dados que você está manipulando e decidir sobre um apropriado meio de armazenamento,

O armazenamento puro na memória é oferecido por muitas soluções de cache para armazenamento em alta velocidade de dados efêmeros, com persistência zero. O armazenamento na memória atende a uma finalidade

semelhante à anterior, com persistência limitada, por exemplo, para evitar a perda das filas de trabalhos entre as reinicializações.



Bancos de dados baseados em disco e logs de confirmação gravam seus dados em disco, como é o caso do MongoDB e SQL. Sistemas de arquivos armazenados em disco para armazenar arquivos regulares, como documentos e imagens, ou arquivos simples, como CSV e JSON.

Ao escolher os dados que deseja armazenar e a maneira como serão armazenados, você precisará considerar o custo de retenção em relação à utilidade contínua. Os dados de clientes de terceiros ficam obsoletos rapidamente, à medida que a demografia e as demandas do público mudam, enquanto os dados de terceiros específicos do seu negócio geralmente são valiosos por um período mais longo.

Para armazenamento de longo prazo com disponibilidade imediata, especialmente de dados analíticos, *data lakes* e *data warehouses* são uma boa escolha.

Atividade Extra

A partir do que foi apresentado neste módulo, principalmente sobre a gestão da Persistência de Dados no desenvolvimento de jogos digitais, o trabalho acadêmico intitulado "**Melodiz 0.2: Jogo para Auxílio no Desenvolvimento da Percepção Musical com Recursos de Gamificação**" (DAMASCENO, GAMA e OLIVEIRA, 2020), disponível no Google Acadêmico, apresenta o processo de atualização de um aplicativo para dispositivos móveis para auxiliar no desenvolvimento da percepção musical. Ao longo do projeto será apresentado que, para o desenvolvimento desse aplicativo, foi adotada e consultada a API da Google Games Services, assim como sua

documentação, para persistência de dados remotos e recursos de gamificação. Para isso, foram empregados os seguintes serviços: Sign Leaderboards e Achievements.

Referência Bibliográfica

COELHO, H. **JPA eficaz: as melhores práticas de persistência de dados em Java**. São Paulo: Casa do Código, 2014.

JANDL JUNIOR, P. **Java: guia do programador: atualizado para Java 8**. 3. ed. São Paulo: Novatec, 2015.

RABIN, S. **Introdução ao Desenvolvimento de Games**. São Paulo: Cengage Learning Brasil, 2012.

SCHATZ, L. R.; VIECELLI, E.; VIGOLO, V. **Banco de Dados - PERSISTE: Framework para persistência de dados isolada à regra de negócios**. Anais V SULCOMP, 2010. Disponível em: https://periodicos.unesc.net/ojs/index.php/sulcomp/article/view/234 >. **(Acesso em: 17/05/2023)**

Atividade Prática

Título da Prática: Persistência de Dados

Objetivos: Compreender os princípios da Persistência de Dados no desenvolvimento de jogos digitais.

Materiais, Métodos e Ferramentas: Browser (navegador) e ferramenta de edição de textos.



Roteiro

1º. Passo) Leia atentamente o texto:

Dados persistentes são dados armazenados em um meio de armazenamento persistente. Um meio de armazenamento persistente (ou não volátil) é um meio onde os dados permanecem intactos depois de terem sido gravados até que sejam substituídos. Isso inclui memória flash (SSDs, pendrives), discos rígidos, fita magnética e mídia óptica.

Qualquer dado que precise ser usado após a conclusão do processo que o gerou deve ser armazenado em um meio de armazenamento persistente. Por exemplo, as faturas são geradas durante o processo de venda de uma empresa a um cliente por meio de sua loja online e precisam ser mantidas após a conclusão da transação de compra.

Ao desenvolver aplicativos, como plataformas de comércio eletrônico, aplicativos de mídia social ou pipelines de dados, você precisa considerar como os diferentes dados gerados e coletados serão utilizados e armazenados.

De acordo com SCHATZ, VIECELLI e VIGOLO (2010), todos os dados que precisam estar disponíveis para uso após a conclusão do processo de seu aplicativo e/ou jogo digital devem ser armazenados em um meio de armazenamento persistente. Isso vai além de apenas manter as transações que ocorrem na loja de itens do seu jogo. Ao longo do ciclo de vida da sua empresa, são gerados dados primários altamente valiosos que podem ter utilidade futura. Você precisa descobrir quais desses dados são mais valiosos e garantir que sejam mantidos, enquanto os dados necessários apenas temporariamente podem ser utilizados e descartados de forma segura.

2º. Passo) Acesse o Google Acadêmico e pesquise o trabalho intitulado de "Framework para Persistência de Dados" (LORENZETTI, 2004) para obser  **sobre a possibilidade da utilização de um Framework para o desenvolvimento da persistência de Dados:**

Um framework de persistência de dados move os dados do programa em sua forma mais natural de e para um armazenamento de dados permanente (a base de dados). Assim, uma estrutura de persistência gerencia o banco de dados e o mapeamento entre o banco de dados e os objetos.

De acordo com LORENZETTI (2004, p. 5), os frameworks para persistência de dados são desenvolvidos tendo como alvo primariamente o conceito de software orientado a objetos, ditando outros aspectos arquiteturais da aplicação. Esse framework definirá o particionamento e organização da aplicação em classes e objetos instanciados a partir delas, bem como o fluxo de dados e controle.

3º. Passo) Desenvolva um relatório técnico que descreva o processo de utilização de frameworks para a persistência de dados:

Os frameworks vêm sendo considerados como tecnologias importantes de reutilização em desenvolvimento de software, pois além de promoverem a reutilização, também promovem a modularidade e extensibilidade do software desenvolvido (LORENZETTI, 2004, p. 99).

A partir desse contexto, você deverá enviar o seu relatório técnico seguindo o formato ABNT e contendo entre 1.000 a 1.500 caracteres (com espaços), desconsiderando as referências bibliográficas. Na montagem desse documento, será permitida a utilização de imagens, gráficos e demais elementos para facilitar a compreensão da sua ideia.

Gabarito comentado:

Um dos componentes essenciais e mais complexos no processo de desenvolvimento de software é o banco de dados. Quando um componente baseado em um tipo de abordagem (por exemplo, orientado a objetos) tenta interagir diretamente com outro objeto que tem suas raízes em outro tipo de abordagem (por exemplo, relacional). Um exemplo disso é o Java Database Connectivity (JDBC). Embora o JDBC forneça um método fácil para acessar diferentes bancos de dados, é basicamente uma API de baixo nível que fornece apenas uma camada fina de abstração. Isso é adequado para pequenos e médios projetos, mas não é adequado para nível empresarial. Com o JDBC, abrir e fechar a conexão envolve muito código. O que é necessário é desenvolver uma estrutura que possa atuar como um mediador entre ambos os elementos.

> A utilização de padrões de projeto é recomendada pela possibilidade de se obter uma estrutura de classes melhor organizada e mais apta à manutenção, pois os padrões representam uma base de conhecimento adquirida por especialistas durante muitos anos. Além disso, a utilização de padrões na construção de um framework diminui o tempo de aprendizado de como utilizar o framework para desenvolver aplicações, uma vez que as pessoas que conhecem os padrões terão maior facilidade em utilizá-lo (LORENZETTI, 2004, p. 99).

Conforme THAMPI e ASHWIN (2007), uma estrutura de persistência move os dados do programa em sua forma mais natural, ou seja, em objetos de memória, para um armazenamento de dados permanente, que é o banco de dados. O framework de persistência gerencia o banco de dados e o mapeamento entre o banco de dados e os objetos. A estrutura de persistência simplifica o processo de desenvolvimento. Existem muitos frameworks de persistência (tanto Open Source e Comercial) no mercado. Hibernate e iBatis são exemplos de frameworks para Java. Por exemplo, o **Hibernate** é um projeto de código aberto que contém recursos

interessantes, como transparência real. Além disso, utiliza o processamento de bytecode para estender a partir dessas classes e implementar persistência. Segue a imagem abaixo do site oficial do Hibernate:

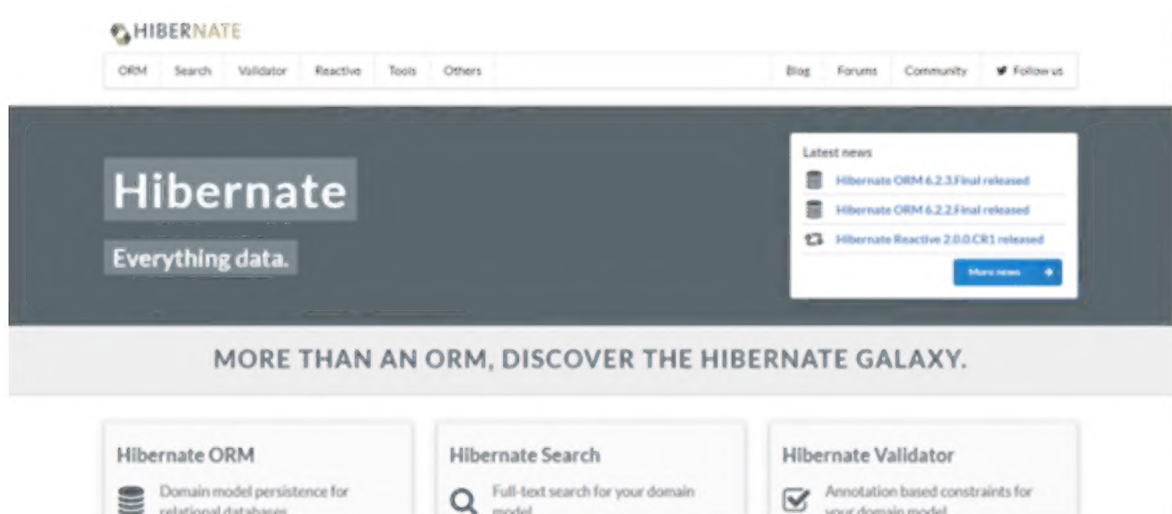


Figura 1 – Reprodução do site oficial do Hibernate.

Fonte: https://hibernate.org/ **(Acesso em 31/05/2023)**

Por fim, podemos apontar que o Hibernate reduz as linhas de código mantendo o próprio mapeamento objeto-tabela e retorna o resultado para o aplicativo na forma de objetos Java. Isso permite que o programador não precise lidar manualmente com os dados persistentes, reduzindo o tempo de desenvolvimento e o custo de manutenção.

Referência Bibliográficas

LORENZETTI, C. **Framework para Persistência de Dados**. Novo Hamburgo, 2004. Disponível em: < http://www.sysrs.com.br/arquivos/principal/MON%20004.415%20L869f.pdf >. **(Acesso em 31/05/2023)**

SCHATZ, L. R.; VIECELLI, E.; VIGOLO, V. **Banco de Dados - PERSISTE: Framework para persistência de dados isolada à regra de negócios**. Anais V

SULCOMP, 2010. Disponível em: < https://periodicos.unesc.net/ojs/index.php/sulcomp/article/view/234 >. **(Ace em 31/05/2023)**

THAMPI, S.; ASHWIN, K. **Performance Comparison of Persistence Frameworks**. 2007. Disponível em: < https://www.researchgate.net/publication/1768886_Performance_Comparison_of_Persistence_Frameworks >. **(Acesso em 31/05/2023)**

Ir para exercício